Moroccan Tournament of Young Mathematicians 2024

Team Name: Infinity and Beyond

Team Members:
TIGUINT Mountassir, ACHAK Hamza, BENTAYEBI Haitam,
GMOUH Yacine, AHIZOUNE Mustapha Amine

Mentor: ELIBBAOUI Mohamed

AUI Supervisor: MIKOU Rym

"Mathematics is a hard thing to love"

December 17, 2024

Contents

1 Amassons les jetons

1.1 Introduction

We successfully gained a strong understanding of the initial problem.We addresed the first three components through a case-by-case / Bash aproach. Afterward, we derived a formula to describe the situation when the variables were real numbers. While this formula was useful, we encountered a limitation: it couldn't be directly applied to the discrete case, where the variables take integer values.

However, during this process, we discovered an interesting property of the optimal solution in the integer case that turned out to be highly useful. This property helped us simplify the problem significantly, enabling us to compute the optimal solution with much greater ease. Essentially, the insight into this property gave us an approach that was efficient and practical for the discrete scenario.

Looking ahead, we explored some additional strategies. Specifically, we considered introducing extra degrees of freedom into the model, which would allow us to test more flexible configurations. we also believe we could potentially find more effective ways to tackle the discrete case. This approach would be beneficial because it would enable us to handle larger values for the parameters N and K—thus extending the scope of our analysis and making the solution applicable to more complex instances of the problem.

1.2 Context

Abla and Brahim have a new game involving a board with 2 cells and 10 tokens each. Abla (Player 1) first places her tokens on the two cells, represented as (a_1, a_2) where $a_1 + a_2 = 10$. Then Brahim (Player 2) observes Abla's placement and places his tokens, represented as (b_1, b_2) with $b_1 + b_2 = 10$. The configuration result is given by:

$$R((a_1, a_2); (b_1, b_2)) = a_1 \cdot (b_1 + b_2) + 2 \cdot a_2 \cdot b_2.$$

Their goals are opposite: Abla maximizes, while Brahim minimizes this result.

1.3 Warm-Up Exercises

1. Abla plays $(5, 5)$, and Brahim responds with $(3, 7)$. What is the result of the configuration in this case? Does $(3, 7)$ allow Brahim to achieve the smallest possible result? If not, what should Brahim play for the result to be the smallest possible, and what is that value?

   Answer: Given the configuration $(a_1, a_2) = (5, 5)$ and $(b_1, b_2) = (3, 7)$, we first calculate the result using the formula:

   $$R((a_1, a_2); (b_1, b_2)) = a_1 \cdot (b_1 + b_2) + 2 \cdot a_2 \cdot b_2$$

   Substituting the values:

   $$R((5, 5); (3, 7)) = 5 \cdot (3 + 7) + 2 \cdot 5 \cdot 7 = 5 \cdot 10 + 2 \cdot 35 = 50 + 70 = 120$$

   To check if $(3, 7)$ allows Brahim to achieve the smallest possible result, we analyze the formula. The term $b_1 + b_2$ is constant because it represents the total allocation of Brahim's resources. Specifically, $b_1 + b_2 = 10$, which means the first term $a_1 \cdot (b_1 + b_2)$ is fixed at $5 \cdot 10 = 50$. Therefore, to minimize the result, Brahim should focus on minimizing the second term, $2 \cdot a_2 \cdot b_2$. Setting $b_2 = 0$ achieves this goal, as any positive value of b_2 increases the result. (We will name this paragraph "Intuitive")

   Thus, the optimal move for Brahim is $(10, 0)$:

   $$R((5, 5); (10, 0)) = 5 \cdot (10 + 0) + 2 \cdot 5 \cdot 0 = 50 + 0 = 50$$

   Therefore, Brahim should play $(10, 0)$ to obtain the smallest possible result of $\boxed{50}$.

Abla plays $(4, 6)$. What is Brahim's best response to minimize R? Calculate the minimum result.

Answer: Brahim should play $(10, 0)$ to minimize R. Proof : Intuitive

Minimum result:

$$R((4, 6); (10, 0)) = 4 \cdot (10) + 2 \cdot 6 \cdot 0 = 40 + 0 = 40$$

Abla notices that when she plays $(5, 5)$, the value of the smallest possible result is different from the one obtained when playing $(4, 6)$. What should Abla play so that the smallest possible result in the final configuration is maximized? What should Brahim play in response, and what is the result of the configuration?

Answer: Abla should play $(10, 0)$ and Brahim should respond by playing $(10, 0)$ to maximize/minimize the result respectively (Intuitive again). The result of the configuration is:

$$R((10, 0); (10, 0)) = 10 \cdot (10 + 0) + 2 \cdot 0 \cdot 0 = 100 + 0 = 100.$$

1.4 New Rules: Inverted Game Value

1. They realize they didn't finish reading the rules. The game's final score involves the inverted game value $V(b_1, b_2)$, defined as:

$$V(b_1, b_2) = \max_{(a_1,a_2)\,:\,a_1+a_2=10} R((a_1, a_2); (b_1, b_2))$$

2.a. Calculate $V(10, 0)$.

Answer: We have:

$$\begin{aligned} V(10, 0) &= \max_{a_1+a_2=10} R((a_1, a_2); (10, 0)) \\ &= \max_{a_1+a_2=10} a_1 \cdot 10 + 2 \cdot a_2 \cdot 0 \\ &= 100. \end{aligned} \tag{1}$$

This maximum is achieved when $(a_1, a_2) = (10, 0)$. (intuitive)

2.b. Calculate $V(5, 5)$.

Answer: We have:

$$\begin{aligned} V(5, 5) &= \max_{a_1+a_2=10} R((a_1, a_2); (5, 5)) \\ &= \max_{a_1+a_2=10} (a_1 + a_2) \cdot 10 \\ &= 100. \end{aligned} \tag{2}$$

This maximum is achieved for all (a_1, a_2) such that $a_1 + a_2 = 10$.

(a) Now, Abla plays $(5, 5)$, and Brahim responds $(3, 7)$. Using:

$$S((a_1, a_2); (b_1, b_2)) = \frac{R((a_1, a_2); (b_1, b_2))}{V(b_1, b_2)}$$

3.a. Determine the score configuration when Abla plays $(5, 5)$ and Brahim responds with $(3, 7)$. Does this allow Brahim to achieve the smallest possible score configuration? If not, what should Brahim play for the smallest score configuration, and what is that score?

Answer:
We have:

$$\begin{aligned} V(3,7) &= \max_{a_1+a_2=10} R((a_1,a_2);(3,7)) \\ &= \max_{a_1+a_2=10} (a_1 \cdot 10 + a_2 \cdot 14) \\ &= 140. \end{aligned} \tag{3}$$

This maximum is achieved when $(a_1, a_2) = (0, 10)$. (because $10 \leq 14$)
Thus, we compute the score function:

$$S((5,5);(3,7)) = \frac{R((5,5);(3,7))}{V(3,7)} = \frac{5 \cdot 10 + 2 \cdot 5 \cdot 7}{140} = \frac{6}{7}.$$

However, this is not the best answer for Brahim to minimize the score. Let's consider (b_1, b_2) as the optimal response for Brahim:
We have:

$$\begin{aligned} S((5,5);(b_1,b_2)) &= \frac{R((5,5);(b_1,b_2))}{V(b_1,b_2)} \\ &= \frac{5 \cdot 10 + 2 \cdot 5 \cdot b_2}{\max_{a_1+a_2=10} (a_1 \cdot 10 + 2 \cdot a_2 \cdot b_2)}. \end{aligned} \tag{4}$$

Case 1:
If $b_2 \in Z \cap [0, 5]$, then:

$$V(b_1, b_2) = \max_{a_1+a_2=10} (a_1 \cdot 10 + 2 \cdot a_2 \cdot b_2).$$

Since $10 \geq 2b_2$, the solution that maximizes the expression is $(a_1, a_2) = (10, 0)$.
Thus,

$$V(b_1, b_2) = 100.$$

Now, we compute the score function:

$$S((5,5);(b_1,b_2)) = \frac{10(5 + b_2)}{100}.$$

This expression reaches its minimum when $b_2 = 0$, yielding:

$$S((5,5);(b_1,b_2)) = \frac{1}{2}.$$

Case 2:
If $b_2 \in Z \cap [6, 10]$, then:

$$V(b_1, b_2) = \max_{a_1+a_2=10} (a_1 \cdot 10 + 2 \cdot a_2 \cdot b_2).$$

Since $10 \leq 2b_2$, the optimal solution is $(a_1, a_2) = (0, 10)$.
Thus,

$$V(b_1, b_2) = 20b_2.$$

Next, we compute the score function:

$$S((5,5); (b_1, b_2)) = \frac{10(5 + b_2)}{20b_2} = \frac{50}{20b_2} + \frac{1}{2}.$$

Since $S((5,5); (b_1, b_2))$ is always greater than or equal to $\frac{1}{2}$, we conclude that:

$$S((5,5); (b_1, b_2)) \geq \frac{1}{2}.$$

Conclusion:
Brahim should play $(10, 0)$ to minimize the score, and the minimal value of $S((5,5); (b_1, b_2))$ is:

$$S((5,5); (10, 0)) = \frac{1}{2}.$$

3.b. After Abla plays $(4, 6)$, what is the best response for Brahim to achieve the smallest possible score? What is that score?

Answer:
We use the same intuition as before.
We have:

$$\begin{aligned} S((4,6); (b_1, b_2)) &= \frac{R((4,6); (b_1, b_2))}{V(b_1, b_2)} \\ &= \frac{4 \cdot 10 + 2 \cdot 6 \cdot b_2}{\max_{a_1+a_2=10} (a_1 \cdot 10 + 2 \cdot a_2 \cdot b_2)}. \end{aligned} \tag{5}$$

Case 1: $b_2 \in Z \cap [0, 5]$:

$$V(b_1, b_2) = 100 \quad \Rightarrow \quad S((4,6); (b_1, b_2)) = \frac{4}{10}.$$

Case 2: $b_2 \in Z \cap [6, 10]$:

$$V(b_1, b_2) = 20b_2 \quad \Rightarrow \quad S((4,6); (b_1, b_2)) = \frac{2}{b_2} + \frac{6}{10} \geq \frac{4}{10}.$$

Conclusion: Brahim should play $(10, 0)$ to minimize the score, and the minimal score is:

$$S((4,6); (10, 0)) = \frac{4}{10}.$$

3.c. Abla notices that the smallest score differs depending on whether she plays $(5,5)$ or $(4,6)$. What should Abla play to achieve the largest possible score in the final configuration? What is Brahim's response, and what is the final score configuration?

Answer:

From the previous questions, we know:

$$S((a_1,a_2);(b_1,b_2)) = \frac{a_1\cdot 10 + 2\cdot a_2\cdot b_2}{\max(100, 20b_2)}.$$

Let S_B represent Brahim's optimal (minimal) score after Abla plays (a_1,a_2). Then:

$$S_B = \min\left(\frac{a_1\cdot 10 + 2\cdot a_2\cdot b_2}{100}, \frac{a_1\cdot 10 + 2\cdot a_2\cdot b_2}{20b_2}\right).$$

The term $\frac{a_1\cdot 10+2\cdot a_2\cdot b_2}{100}$ is minimized for:

$$(b_1,b_2) = (10,0).$$

The term $\frac{a_1\cdot 10+2\cdot a_2\cdot b_2}{20b_2}$ simplifies to:

$$\frac{1}{10}\left(a_2 + \frac{100a_1}{20b_2}\right),$$

and is minimized for:

$$(b_1,b_2) = (0,10).$$

Thus, Brahim's response must be either:

$$(b_1,b_2) = (10,0) \quad \text{or} \quad (b_1,b_2) = (0,10),$$

The minimal score is determined accordingly.

$$S_B = \min\left(\frac{a_1\cdot 10}{100}, \frac{a_1\cdot 10 + 2\cdot a_2\cdot 10}{20\cdot 10}\right).$$

This simplifies further to:

$$S_B = \min\left(\frac{a_1}{10}, \frac{a_1+2a_2}{20}\right) = \frac{1}{20}\min\left(2a_1, 20-a_1\right).$$

Let S_A represent Abla's optimal (maximal) score:

$$S_A = \frac{1}{20}\max_{0\le a_1\le 10}\left(\min\left(2a_1, 20-a_1\right)\right).$$

Since $2a_1$ increases while $20-a_1$ decreases, the maximum of $\min(2a_1, 20-a_1)$ occurs when:

$$2a_1 = 20 - a_1.$$

Solving this, we find:

$$a_1 = \frac{20}{3} \approx 6.66.$$

Thus, Abla's optimal choice is either $a_1 = 6$ or $a_1 = 7$.
- For $a_1 = 7$, $\min(2 \cdot 7, 20 - 7) = 13$.
- For $a_1 = 6$, $\min(2 \cdot 6, 20 - 6) = 12$.
Since $13 > 12$, Abla's optimal move is $(a_1, a_2) = (7, 3)$.
Brahim's response should be $(b_1, b_2) = (0, 10)$, resulting in the score:

$$S((7,3);(0,10)) = \frac{13}{20}.$$

In Conclusion Abla should play $(7, 3)$ to achieve the largest possible score, Brahim should respond with $(0, 10)$, and the final score configuration is:

$$S((7,3);(0,10)) = \frac{13}{20}.$$

1.5 Generalization

The game offers extensions with K slots (instead of 2) numbered $1, \ldots, K$ and N tokens (instead of 10) per player. In this case, a configuration is given by $(a_1, \ldots, a_K)$ and $(b_1, \ldots, b_K)$, and the result of a configuration is generally given by the formula:

$$R((a_1, \ldots, a_K); (b_1, \ldots, b_K)) = \sum_{i=1}^{K} i \cdot a_i \cdot (b_i + b_{i+1} + \cdots + b_K).$$

We generalize in the same way the definitions of $V(b_1, \ldots, b_K)$ and $S((a_1, \ldots, a_K); (b_1, \ldots, b_K))$.

1.6 Formula : generalized to the real numbers

The goal is to show that Brahim should set all the b_i's to zero except for one, which takes the value N, leading to the formula for S_A:

$$S_A = \frac{1}{N} \max_{a_1 + a_2 + \cdots + a_K = N} \min_{1 \le l \le K} \left(\frac{1}{l} \cdot \sum_{i=1}^{l} i a_i \right).$$

Proof

First, define the value V as follows, using linearity:

$$V(b_1, \ldots, b_K) = N \cdot \max_{1 \le l \le K} l(b_l + b_{l+1} + \cdots + b_K).$$

The score function S is given by:

$$S((a_1, \ldots, a_K); (b_1, \ldots, b_K)) = \frac{1}{N} \cdot \frac{\sum_{i=1}^{K} i a_i (b_i + b_{i+1} + \cdots + b_K)}{\max_{1 \le l \le K} l(b_l + b_{l+1} + \cdots + b_K)}.$$

Let $(a_1, \ldots, a_K)$ be a fixed K-tuple, and let S_B represent Brahim's optimal (minimal) score after Abla plays $(a_1, \ldots, a_K)$. Then:

$$S_B = \frac{1}{N} \min_{1 \le l \le K} \left(\frac{\sum_{i=1}^{K} i a_i (b_i + b_{i+1} + \cdots + b_K)}{l(b_l + b_{l+1} + \cdots + b_K)} \right).$$

We will now show that for any $1 \leq l \leq K$, the expression

$$\frac{\sum_{i=1}^{K} ia_i(b_i + b_{i+1} + \cdots + b_K)}{l(b_l + b_{l+1} + \cdots + b_K)}$$

is minimized when $b_l = N$ and $b_j = 0$ for $j \neq l$.

Proof by Contradiction

Suppose, by way of contradiction (BWOC), that there exists an optimal set $(b_1, \ldots, b_K)$ such that there exists $j \neq l$ with $b_j > 0$.

Case 1: $j > l$ Consider the set $(c_1, \ldots, c_K)$ defined as:

$$(c_1, \ldots, c_K) = (b_1, \ldots, b_l + \epsilon, \ldots, b_j - \epsilon, \ldots, b_K),$$

where $\epsilon > 0$ is a small perturbation.
We have:

$$l(c_l + \cdots + c_K) = l(b_l + \cdots + b_K),$$

and:

$$\sum_{i=1}^{l} ia_i(c_i + c_{i+1} + \cdots + c_K) = \sum_{i=1}^{l} ia_i(b_i + b_{i+1} + \cdots + b_K).$$

However :

$$\sum_{i=l+1}^{K} ia_i(c_i + c_{i+1} + \cdots + c_K) = \sum_{i=l+1}^{K} ia_i\big(b_i + b_{i+1} + \cdots + (b_j - \epsilon) + \cdots + b_K\big) < \sum_{i=l+1}^{K} ia_i(b_i + b_{i+1} + \cdots + b_K).$$

Summing and dividing, we get:

$$\frac{\sum_{i=1}^{K} ia_i(c_i + c_{i+1} + \cdots + c_K)}{l(c_l + c_{l+1} + \cdots + c_K)} < \frac{\sum_{i=1}^{K} ia_i(b_i + b_{i+1} + \cdots + b_K)}{l(b_l + b_{l+1} + \cdots + b_K)}.$$

This contradicts the minimality of $(b_1, \ldots, b_K)$.

Case 2: $j < l$ Consider the set $(c_1, \ldots, c_K)$ defined as:

$$(c_1, \ldots, c_K) = (b_1, \ldots, b_j - \epsilon, \ldots, b_l + \epsilon, 0, 0, \ldots, 0),$$

where $\epsilon > 0$.
Since we proved earlier that for all $j > l$, $b_j = 0$ in an optimal set, we focus on $j < l$.
We have:

$$l(c_l + \cdots + c_K) = l(b_l + \epsilon),$$

and:

$$\sum_{i=1}^{j} ia_i(c_i + c_{i+1} + \cdots + c_K) = \sum_{i=1}^{j} ia_i(b_i + b_{i+1} + \cdots + b_K).$$

We also have:

$$\sum_{i=j+1}^{l} ia_i(c_i + c_{i+1} + \cdots + c_K) = \epsilon \sum_{i=j+1}^{l} ia_i + \sum_{i=j+1}^{l} ia_i(b_i + b_{i+1} + \cdots + b_K).$$

Thus:

$$\frac{\sum_{i=1}^{K} ia_i(c_i + c_{i+1} + \cdots + c_K)}{l(c_l + c_{l+1} + \cdots + c_K)} = \frac{\epsilon \sum_{i=j+1}^{l} ia_i + \sum_{i=1}^{K} ia_i(b_i + b_{i+1} + \cdots + b_K)}{l(b_l + \epsilon)}.$$

To obtain the desired contradiction, we need to show:

$$\frac{\sum_{i=1}^{K} ia_i(b_i + b_{i+1} + \cdots + b_K)}{l(b_l + b_{l+1} + \cdots + b_K)} > \frac{\epsilon \sum_{i=j+1}^{l} ia_i + \sum_{i=1}^{K} ia_i(b_i + b_{i+1} + \cdots + b_K)}{l(b_l + \epsilon)}.$$

Since $b_{l+1} = b_{l+2} = \cdots = 0$, this simplifies to:

$$\frac{\sum_{i=1}^{l} ia_i(b_i + b_{i+1} + \cdots + b_l)}{lb_l} > \frac{\epsilon \sum_{i=j+1}^{l} ia_i + \sum_{i=1}^{l} ia_i(b_i + b_{i+1} + \cdots + b_l)}{l(b_l + \epsilon)}.$$

Clearing the denominator, this reduces to:

$$\sum_{i=1}^{l} ia_i(b_i + b_{i+1} + \cdots + b_l) > b_l \cdot \sum_{i=j+1}^{l} ia_i,$$

which is clearly true.

Conclusion

To minimize Brahim's score, all b_j's must be zero except for b_l, which equals N. Consequently, Brahim's score is given by:

$$S_B = \frac{1}{N} \min_{1 \le l \le K} \left(\frac{\sum_{i=1}^{K} ia_i(b_i + b_{i+1} + \cdots + b_K)}{l(b_l + b_{l+1} + \cdots + b_K)} \right) = \frac{1}{N} \min_{1 \le l \le K} \left(\frac{\sum_{i=1}^{l} ia_i N}{lN} \right) = \frac{1}{N} \min_{1 \le l \le K} \left(\frac{1}{l} \sum_{i=1}^{l} ia_i \right).$$

Let S_A represent Abla's optimal (maximal) score. We obtain:

$$S_A = \frac{1}{N} \max_{a_1 + a_2 + \cdots + a_K = N} \min_{1 \le l \le K} \left(\frac{1}{l} \sum_{i=1}^{l} ia_i \right).$$

Remark

It is important to note that the expression $\max_{1 \le i \le K} i(b_i + \cdots + b_K)$ will still be attained at the index l. This is because, after the new optimization, if we set $x_i = i(b_i + \cdots + b_K)$, then for $l < i$, we have $x_i = 0$, and for $i < l$, we have $x_i = iN$, which is clearly smaller than $x_l = lN$.

Alternate Proof

We start with the goal of proving that Brahim should set all the b_i's at zero, except one of them, which should take the value N. This will lead to the formula for S_A given by:

$$S_A = \frac{1}{N} \max_{a_1+a_2+\cdots+a_K=N} \min_{1\leq l\leq K} \left(\frac{1}{l} \cdot \sum_{i=1}^{l} ia_i \right).$$

Proof

First of all, we know that:

$$V(b_1, \ldots, b_K) = N \max_{1\leq l\leq K} i(b_i + \cdots + b_K)$$

by linearity.
Then,

$$S((a_1, \ldots, a_K); (b_1, \ldots, b_K)) = \frac{1}{N} \frac{\sum_{i=1}^{l} ia_i(b_i + \cdots + b_K)}{\max_{1\leq l\leq K} i(b_i + \cdots + b_K)}.$$

Let $(b_1, \ldots, b_K)$ be a K-tuple, and denote by j the index such that:

$$\max_{1\leq i\leq K} i(b_i + \cdots + b_K) = j(b_j + \cdots + b_K).$$

Now, let's focus on the best choice for Brahim in this particular case (where by "best choice" we mean the choice that minimizes the final score).

First Optimization

If we replace $b_{j+1}, \ldots, b_K$ all by 0 and set b_j to:

$$b_j + b_{j+1} + \cdots + b_K,$$

we go from the initial K-tuple $(b_1, \ldots, b_K)$ to the new K-tuple:

$$(b_1, \ldots, b_{j-1}, b_j + b_{j+1} + \cdots + b_K, 0, 0, \ldots, 0).$$

From now on, to avoid introducing new variables, consider the new K-tuple $(b_1, \ldots, b_{j-1}, b_j + b_{j+1} + \cdots + b_K, 0, 0, \ldots, 0)$.
The sum of the b_i's, as well as $\max_{1\leq i\leq K} i(b_i + \cdots + b_K)$, remain constant. A quick way to see this is to notice that $i(b_i + \cdots + b_K)$ becomes 0 for $j + 1 \leq i \leq K$ and does not change for $1 \leq i \leq j$.
Also, $\sum_{i=1}^{K} ia_i(b_i + \cdots + b_K)$ clearly decreases by exactly $\sum_{i=j+1}^{K} ia_i(b_i + \cdots + b_K)$.
Now, looking at the definition of the score, we can see that this new K-tuple of b_i's gives a smaller score. After this smoothing/optimization, the score can be rewritten as:

$$S((a_1, \ldots, a_K); (b_1, \ldots, b_K)) = \frac{1}{N} \frac{\sum_{i=1}^{K} ia_i(b_i + \cdots + b_K)}{\max_{1\leq l\leq K} i(b_i + \cdots + b_K)}.$$

This simplifies to:

$$S((a_1, \ldots, a_K); (b_1, \ldots, b_K)) = \frac{1}{Nj} \left(\sum_{i=1}^{j} ia_i + \sum_{i=1}^{j-1} \frac{b_i + \cdots + b_{j-1}}{b_j} \right). \qquad (*)$$

Second Optimization

In this step, we use the same idea as in the previous one. If we replace $b_{j-1}, \ldots, b_1$ all by 0 and set b_j to:

$$b_j + b_{j-1} + \cdots + b_1,$$

we go from the K-tuple $(b_1, \ldots, b_{j-1}, b_j + b_{j+1} + \cdots + b_K, 0, 0, \ldots, 0)$ to:

$$(0, \ldots, 0, b_1 + \cdots + b_j + \cdots + b_K, 0, 0, \ldots, 0).$$

The sum of the b_i's remains constant, and the maximum value $\max_{1 \leq i \leq K} i(b_i + \cdots + b_K)$ is still attained at the index j.
Using the equality (*) from earlier, we can easily see that this second smoothing/optimization will make the overall score decrease. Specifically: - $\sum_{i=1}^{j} ia_i$ stays constant. - $\sum_{i=1}^{j-1} \frac{b_i + \cdots + b_{j-1}}{b_j}$ becomes 0, which clearly decreases the score.
After these two optimizations, we are left with the K-tuple of b_i's: $(0, \ldots, N, \ldots, 0)$, with the N in the j-th position. This configuration yields a smaller score than the initial one.

Conclusion

Brahim should set all the b_i's to zero, except for one index, which should take the value N. Let's assume he chooses the index h.
Recalling the formula for the score:

$$S((a_1, \ldots, a_K); (b_1, \ldots, b_K)) = \frac{1}{N} \frac{\sum_{i=1}^{h} ia_i}{h},$$

to minimize the score, Brahim should choose the index h that minimizes this result. Therefore, the minimal value of the score is:

$$S((a_1, \ldots, a_K); (b_1, \ldots, b_K)) = \min_{1 \leq i \leq K} \left(\frac{1}{N} \frac{\sum_{i=1}^{h} ia_i}{h} \right).$$

Finally, recall that Alba wants to maximize the total score, leading to the following formula for S_A:

$$S_A = \frac{1}{N} \max_{a_1 + a_2 + \cdots + a_K = N} \min_{1 \leq l \leq K} \left(\frac{1}{l} \cdot \sum_{i=1}^{l} ia_i \right).$$

1.7 Natural numbers

Important Note

This section is meant to be read after understanding the following section about the continuous case. The reasoning and results presented here cannot be fully understood otherwise.

1.7.1 Floor or ceiling ?

We avoided calculating all possible configurations directly because the number of configurations grows exponentially with K and N. For example, with $K = 10$ and $N = 100$, there would be roughly 380 million configurations, but using our method, we reduced the number of possibilities to $2^{10} = 1024$. To do this, we used the floor/ceiling approach to reduce the number of configurations we needed to explore. This approach is motivated by the fact that, in the continuous case, the optimal solution occurs when the values a_i are balanced. Specifically, for each l, the terms

$$\frac{1}{l} \cdot \sum_{i=1}^{l} i a_i$$

should be equal. By using the floor and ceiling of the values, we efficiently approximate the optimal solution, ensuring that the terms are balanced while avoiding an exhaustive search over all configurations.

Claim

Alba should maximize

$$S_A = \frac{1}{N} \max_{a_1+a_2+\cdots+a_K=N} \min_{1\le l\le K} \left(\frac{1}{l} \cdot \sum_{i=1}^{l} i a_i \right)$$

by setting

$$a_i = \lfloor \frac{N}{iH_k} \rfloor \quad \text{or} \quad a_i = \lceil \frac{N}{iH_k} \rceil.$$

Proof:

Assume, for the sake of contradiction, that Alba achieves a maximal score by setting some a_i that do not satisfy the stated properties.

Case 1: There exists an a_{i_0} such that

$$a_{i_0} \le \lfloor \frac{N}{i_0 H_k} \rfloor - 1.$$

Since $\sum_{i=1}^{K} a_i = N$, there must exist an index j such that

$$a_j \ge \lfloor \frac{N}{jH_k} \rfloor + 1.$$

If Alba replaces a_{i_0} with $a_{i_0} + 1$ and a_j with $a_j - 1$, the value of S_A would increase, contradicting the maximality of S_A.

Case 2: There exists an a_{i_0} such that

$$a_{i_0} \geq \lceil \frac{N}{i_0 H_k} \rceil + 1.$$

Since $\sum_{i=1}^{K} a_i = N$, there must exist an index l such that

$$a_l \leq \lfloor \frac{N}{l H_k} \rfloor.$$

If Alba replaces a_{i_0} with $a_{i_0} - 1$ and a_l with $a_l + 1$, the value of S_A would increase, contradicting the maximality of S_A.

Conclusion:

To achieve a maximal value of S_A, the a_i's should satisfy

$$a_i = \lfloor \frac{N}{i H_k} \rfloor \quad \text{or} \quad a_i = \lceil \frac{N}{i H_k} \rceil.$$

In order to solve the question for the given values we will use a C++ program that tries all the configurations of floor/ceiling for each a_i with the condition that the sum is N

The program first computes the harmonic number H_n for a given n using the formula:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}$$

Next, it generates possible values for each sequence element by dividing the target sum by $i \times H_n$ (where i is the index of the element) and computing both the floor and ceiling of these base values. The valid combinations of these floor and ceiling values are explored using bitmasking, where each bit in a bitmask determines whether the floor or ceiling value is selected for a given element. This ensures all possible valid sequences are considered, and only those whose sum matches the target sum are retained.

For each valid sequence, the program computes the objective function for each i from 1 to the number of elements in the sequence. The objective function is:

$$f(\text{sequence}) = \frac{1}{N} \times \frac{1}{i} \sum_{k=1}^{i} k \cdot a_k$$

Where a_k is the k-th element of the sequence, and i represents the number of elements used in the weighted sum. The program minimizes $f(\text{sequence})$ for each possible i and identifies the sequence that maximizes the minimum value of this function across all configurations. The sequence and its corresponding minimized value are then output.

1.7.2 Code

```cpp
double harmonic_number(int n) {
    double sum = 0.0;
    for (int i = 1; i <= n; ++i) {
        sum += 1.0 / i;
    }
    return sum;
}
double compute_f(vector<int>& sequence, int i) {
    double sum = 0.0;
    for (int k = 1; k <= i; ++k) {
        sum += k * sequence[k - 1];
    }
    return (1.0 / i) * sum;
}
int main() {
    int n, num_elements, target_sum;
    cin >> n;
    cin >> num_elements;
    cin >> target_sum;

    double H_n = harmonic_number(n);

    vector<pair<int, int>> values;
    for (int i = 1; i <= num_elements; ++i) {
        double base_value = static_cast<double>(target_sum) / (i * H_n);
        values.push_back({floor(base_value), ceil(base_value)});
    }

    vector<int> best_sequence;
    double best_min_value = -numeric_limits<double>::infinity();
    int best_min_i = -1;

    for (int mask = 0; mask < (1 << num_elements); ++mask) {
        vector<int> sequence;
        for (int j = 0; j < num_elements; ++j) {
            if (mask & (1 << j)) {
                sequence.push_back(values[j].second);
            } else {
                sequence.push_back(values[j].first);
            }
        }

        if (accumulate(sequence.begin(), sequence.end(), 0) == target_sum) {
            double min_value = numeric_limits<double>::infinity();
            int min_i = -1;

            for (int i = 1; i <= num_elements; ++i) {
                double f_value = compute_f(sequence, i);
                if (f_value < min_value) {
                    min_value = f_value;
                    min_i = i;
                }
            }

            cout << "Valid sequence: ";
            for (int val : sequence) {
                cout << val << " ";
```

```
            }
            cout << "\nMinimized i: " << min_i << ", Min value: "
                 << fixed << setprecision(6) << min_value / target_sum<< endl;

            if (min_value > best_min_value) {
                best_min_value = min_value;
                best_min_i = min_i;
                best_sequence = sequence;
            }
        }
    }

    cout << "\nSequence that maximizes the minimized value: ";
    for (int val : best_sequence) {
        cout << val << " ";
    }
    cout << "\nMinimized value: " << fixed << setprecision(6)
         << best_min_value / target_sum << ", i: " << best_min_i << endl;

    return 0;
}
```

1.7.3 Answers to section 4

```
> Running optimization for K=3, N=11...
Optimal set: (6, 3, 2)
Score: 6/11 = 0.5454545...

> Running optimization for K=5, N=20...
Optimal sets: (9, 4, 3, 2, 2) or (9, 4, 3, 3, 1)
Score: 17/40 = 0.425

> Running optimization for K=10, N=100...
Optimal set: (34, 17, 12, 8, 7, 6, 5, 4, 4, 3)
Score: 17/50 = 0.34
```

1.8 Real numbers

For each of the following questions, find what Abla must play in order to maximize the score, what Brahim must answer to minimize the score, and the value of the score in this case for K's subsequent choices. (As you did for question 4). Note that we do not give a value of N since now we impose the constraints: $a_1 + ... + a_K = 1$and $b_1 + ... + b_K = 1$

$$(a) K = 3$$

$$(b) K = 5$$

$$(c) K = 10$$

$$(d) K = 100$$

1.8.1 Solving the problem for an arbitrary value of K

Lets solve the problem for the case when

$$a_1 + a_2 + ... + a_K = 1$$

$$b_1 + b_2 + ... + b_k = 1$$

define S_A as the best score that Alba can get no matter how Brahim plays
we have that

$$S_A = \max_{a_1+a_2+....+a_K=1} \min_{1\leq l\leq K} \left(\frac{1}{l} \cdot \sum_{i=1}^{l} i a_i \right)$$

I claim that $S_A = \frac{1}{H_K}$ where $H_k = 1 + \frac{1}{2} + ... + \frac{1}{k}$ Lets proof this result :

Part 1: $S_A \geq \frac{1}{H_K}$

PROOF: consider the following construction for Alba $a_i = \frac{1}{i \cdot H_i}$ for all $1 \leq i \leq K$ then clearly:

$$S_A = \max_{a_1+a_2+....+a_K=1} \min_{1\leq l\leq K} \left(\frac{1}{l} \cdot \sum_{l=1}^{l} i a_i \right) = \frac{1}{H_k}$$

and

$$\sum_{i=1}^{K} a_i = 1$$

so we conclude that $S_A \geq \frac{1}{H_K}$

Part 2: $S_A \leq \frac{1}{H_K}$

PROOF : I need to proof that for any choice of $(a_1, ..., a_k)$ I can find an $1 \leq i_0 \leq k$ such that

$$\frac{1}{i_0} \cdot \sum_{i=1}^{i_0} i a_i \leq \frac{1}{H_K}$$

now suppose BWOC that there exist a k-uplet $(a_1, ..., a_k)$ such that for all $1 \leq i \leq k$

$$\frac{1}{i} \cdot \sum_{s=1}^{i} sa_s > \frac{1}{H_K}(*)$$

lets proof by induction on $1 \leq n \leq k$ that

$$a_1 + ... + a_n > \frac{H_n}{H_K}$$

the base case $n = 1$ follow from our supposition (*) for $i = 1$ Now suppose we have proved it all the way up to n-1 ie $\left(\forall 1 \leq i \leq n-1 / a_1 + ... + a_i > \frac{H_i}{H_K}\right)$ then by our original supposition (*) :

$$\frac{1}{n} \cdot \sum_{s=1}^{n} sa_s > \frac{1}{H_K}$$

and by the induction hypothesis we have the following n-1 inequalities :

$$\forall 1 \leq i \leq n-1$$

$$\frac{1}{n} \cdot (a_1 + ... + a_i) > \frac{1}{n} \cdot \frac{H_i}{H_K}$$

summing up these n inequalities we get that

$$a_1 + ... + a_n > \frac{1}{H_K} + \frac{1}{n} \cdot \sum_{i=1}^{n-1} \frac{H_i}{H_k}$$

$$= \frac{1}{H_K} \cdot \left(1 + \frac{1}{n} \cdot \sum_{i=1}^{n-1} H_i\right)$$

$$= \frac{1}{H_K} \cdot \left(1 + \frac{1}{n} \cdot \sum_{i=1}^{n-1} \sum_{j=1}^{i} \frac{1}{j}\right)$$

$$= \frac{1}{H_K} \cdot \left(1 + \frac{1}{n} \cdot \sum_{j=1}^{n-1} \frac{n-j}{j}\right)$$

$$= \frac{1}{H_K} \cdot \left(1 + H_{n-1} - \frac{n-1}{n}\right)$$

$$= \frac{H_n}{H_K}$$

wich completes our induction . and so $\forall 1 \leq n \leq k$:

$$a_1 + ... + a_n > \frac{H_n}{H_K}$$

replacing n by K we get

$$1 = a_1 + ... + a_K > \frac{H_K}{H_K} = 1$$

which is absurd and contradicts our original supposition (*)

So now we have proved that for any choice of $(a_1, ..., a_K)$ we can find an $1 \leq i_0 \leq K$ such that

$$\frac{1}{i_0} \cdot \sum_{i=1}^{i_0} i a_i \leq \frac{1}{H_K}$$

and hence $S_A \leq \frac{1}{H_k}$ wich concludes the proof of Part 2

Part 3: Conclusion $S_A = \frac{1}{H_K}$

the conclusion follow directly from Part 1 and Part 2 and so in order to maximize his score Alba need to play $a_i = \frac{1}{i \cdot H_k}$ for all $1 \leq i \leq K$ and Brahim should respond with$(b_1, ..., b_k) = (1, 0, ..., 0)$

1.8.2 Answers for section 5

$$(a) K = 3$$

Alba need to play $a_i = \frac{1}{i \cdot H_3}$ for all $1 \leq i \leq 3$ and Brahim should respond with$(b_1, b_2, b_3) = (1, 0, 0)$ to obtain the score : $S_A = \frac{1}{H_3} = \frac{6}{11}$

$$(b) K = 5$$

Alba need to play $a_i = \frac{1}{i \cdot H_5}$ for all $1 \leq i \leq 5$ and Brahim should respond with$(b_1, ..., b_5) = (1, ..., 0)$ to obtain the score : $S_A = \frac{1}{H_5} = \frac{60}{137}$

$$(c) K = 10$$

Alba need to play $a_i = \frac{1}{i \cdot H_{10}}$ for all $1 \leq i \leq 10$ and Brahim should respond with$(b_1, ..., b_{10}) = (1, ..., 0)$ to obtain the score : $S_A = \frac{1}{H_{10}} = \frac{2520}{7381}$

$$(d) K = 100$$

Alba need to play $a_i = \frac{1}{i \cdot H_{100}}$ for all $1 \leq i \leq 100$ and Brahim should respond with$(b_1, ..., b_{100}) = (1, ..., 0)$ to obtain the score : $S_A = \frac{1}{H_{100}} = \frac{956655811982011}{4962534851194762}$

1.9 Research Tracks

1.9.1 A New problem ?

our initial definition is

$$R((a_1, .., a_K); (b_1, ..., b_K)) = \sum_{i=1}^{K} i a_i (b_i + ... + b_K)$$

but why not replace the i with an arbitrary weights and see how does the optimal values change namely if we set

$$R((a_1, .., a_K); (b_1, ..., b_K) = \sum_{i=1}^{K} w_i a_i (b_i + ... + b_K)$$

for some w_i being arbitrarly positive reals what is the condition on the w_i's such that the problem is still solvable

1.9.2 Too Complex

In the discreate case in order to compute the score we were forced to run a programm thats checks a lot cases , and as discussed in the previous sections , we were able to reduce drastically the number of possibilities . the question is can we go further (find a more efficient algo) and if yes , by how much ?

Lemma:

- the sequence a_i satisfies:

$$a_i = \lfloor y_i \rfloor \quad \text{or} \quad a_i = \lceil y_i \rceil,$$

where:

$$y_i = \frac{N}{iH_k},$$

- We also know that there sum is equal , i.e.:

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} y_i = N$$

- We will prove that the number of times we choose the *ceiling* (i.e., $a_i = \lceil y_i \rceil$) is fixed.

Step 1: Sum Condition

We know that:

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} y_i.$$

Now, consider the difference between y_i and a_i for each i:

$$y_i = \lfloor y_i \rfloor + \{y_i\},$$

where $\{y_i\}$ is the fractional part of y_i. The value $y_i - \lfloor y_i \rfloor = \{y_i\}$ is the fractional part of y_i, and it lies in the interval $[0, 1)$.
So for each i: - If $a_i = \lfloor y_i \rfloor$, the contribution to the sum is $\lfloor y_i \rfloor$. - If $a_i = \lceil y_i \rceil$, the contribution to the sum is $\lceil y_i \rceil = \lfloor y_i \rfloor + 1$.
The sum of the a_i's can be written as:

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} \lfloor y_i \rfloor + (\text{number of ceilings}).$$

The sum of the y_i's is:

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \left(\lfloor y_i \rfloor + \{y_i\} \right).$$

Thus, the condition becomes:

$$\sum_{i=1}^{n} \lfloor y_i \rfloor + (\text{number of ceilings}) = \sum_{i=1}^{n} \left(\lfloor y_i \rfloor + \{y_i\} \right).$$

Simplifying this, we get:

$$(\text{number of ceilings}) = \sum_{i=1}^{n} \{y_i\}.$$

Step 2: Conclusion

Thus, the number of times we choose the ceiling function $a_i = \lceil y_i \rceil$ is fixed because it exactly equals the sum of the fractional parts $\sum_{i=1}^{n} \{y_i\}$, which is a fixed number once N and H_k are given. Therefore, the number of ceilings is determined by the fractional parts of y_i and is **fixed**.

Methods to compute the right sequence of ai's

Brute Force Computation (Stars and Bars)

The brute force approach to this problem involves calculating all possible ways to partition N into K parts.which can be seen as a combinatorial task. The combinatorial formula for this is:

$$\binom{N+K-1}{K-1}.$$

Thus, the complexity of the brute force method is $O(N^{K-1})$, which grows polynomially with N for small K, but becomes inefficient as N increases.

Program 2 (Exponential Growth)

Program 2 explores all 2^K combinations of floor and ceiling choices for each y_i. The time complexity is:

$$O(2^K).$$

For small K, this exponential growth is manageable, but as K becomes large, the number of combinations increases dramatically, making the method inefficient for large K.

Program 3 (Slower Exponential Growth)

we us the info that the number of times we choose 'ceilling' is fixed and we can compute it . Program 3 improves upon Program 2 by reducing the cases to at most The binomial coefficient $\binom{K}{\lfloor K/2 \rfloor}$ wich grows exponentially with K, but slower than 2^K. For large K, it can be approximated as:

$$\binom{K}{\lfloor K/2 \rfloor} \sim \frac{2^K}{\sqrt{K}}.$$

Thus, the growth is exponential, but reduced by a factor of $\sqrt{K}$.

Comparison of Programs

1. For Large N and Small K

- Brute Force (Stars and Bars): The complexity $O(N^{K-1})$ grows polynomially with N. For small K, brute force remains feasible but becomes inefficient as N increases. - Program 2 ($O(2^K)$): For small K, the complexity $O(2^K)$ is manageable. As N grows large, this method works efficiently. - Program 3($O\left(\frac{2^K}{\sqrt{K}}\right)$): More efficient than Program 2 for large N due to slower exponential growth.

2. For Small N and Large K

- Brute Force (Stars and Bars): The complexity $O(K^{N-1})$ wich is polynomial in K. - Program 2 ($O(2^K)$): For large K, Program 2 becomes computationally expensive, even if N is small. - program 3 ($O\left(\frac{2^K}{\sqrt{K}}\right)$): The most efficient for large K, as the complexity grows slower than Program 2.

Conclusion

- For large N with small K: The best choices are Program 2 or Program 3 with Program 3 being the most practical for small K.
- For small N with large K: Program 1 is the most efficient as it has a polynomial growth .

2 Quoi pour qui?

2.1 Introduction

This work addresses the challenge of fair resource allocation under d-concave utility functions, which model diminishing returns. Our primary contribution is the development of a polynomial-time algorithm for non-monotone allocation problems, achieving an approximation ratio of 2 for monotone cases, and $1/4$ for non-monotone cases. The algorithm leverages the power of expectation maximization to ensure near-optimal results in complex scenarios. We also explore the implications of d-concavity in these allocations, providing tight bounds that highlight the efficiency of our approach in a variety of combinatorial settings, such as matroids. By extending existing techniques, we bridge the gap between optimality and fairness in real-world applications, making significant strides in both theory and practice.

Fatima and Mohamed face the challenge of distributing gifts to their children optimally. Given the happiness functions f_i for each child, we aim to maximize the total happiness by partitioning the gift set optimally.

2.2 Solutions to the Questions

2.2.1 Case 1: Optimal Distribution of Gifts

Given the happiness tables for Ismail and Omar, we solve for optimal distributions by checking each partition.

Case 1 For the first table, we want to maximize:

$$f_{\text{Ismail}}(S_{\text{Ismail}}) + f_{\text{Omar}}(S_{\text{Omar}})$$

where $S_{\text{Ismail}} \cup S_{\text{Omar}} = \{1, 2, 3\}$ and $S_{\text{Ismail}} \cap S_{\text{Omar}} = \emptyset$.
We evaluate the partitions:

- $S_{\text{Ismail}} = \{1\}, S_{\text{Omar}} = \{2, 3\}$

$$f_{\text{Ismail}}(\{1\}) + f_{\text{Omar}}(\{2, 3\}) = 2 + 5 = 7$$

- $S_{\text{Ismail}} = \{2\}, S_{\text{Omar}} = \{1, 3\}$

$$f_{\text{Ismail}}(\{2\}) + f_{\text{Omar}}(\{1, 3\}) = 3 + 5 = 8$$

- $S_{\text{Ismail}} = \{3\}, S_{\text{Omar}} = \{1, 2\}$

$$f_{\text{Ismail}}(\{3\}) + f_{\text{Omar}}(\{1, 2\}) = 0 + 4 = 4$$

- $S_{\text{Ismail}} = \{1, 2\}, S_{\text{Omar}} = \{3\}$

$$f_{\text{Ismail}}(\{1, 2\}) + f_{\text{Omar}}(\{3\}) = 4 + 3 = 7$$

- $S_{\text{Ismail}} = \{1, 3\}, S_{\text{Omar}} = \{2\}$

$$f_{\text{Ismail}}(\{1, 3\}) + f_{\text{Omar}}(\{2\}) = 2 + 4 = 6$$

- $S_{\text{Ismail}} = \{2, 3\}, S_{\text{Omar}} = \{1\}$

$$f_{\text{Ismail}}(\{2, 3\}) + f_{\text{Omar}}(\{1\}) = 3 + 2 = 5$$

The optimal distribution is $S_{\text{Ismail}} = \{2\}$ and $S_{\text{Omar}} = \{1, 3\}$ with a total happiness of 8.

Case 2 For the second table, we evaluate similarly:

- $S_{\text{Ismail}} = \{1\}, S_{\text{Omar}} = \{2,3\}$

$$f_{\text{Ismail}}(\{1\}) + f_{\text{Omar}}(\{2,3\}) = 1 + 6 = 7$$

- $S_{\text{Ismail}} = \{2\}, S_{\text{Omar}} = \{1,3\}$

$$f_{\text{Ismail}}(\{2\}) + f_{\text{Omar}}(\{1,3\}) = 2 + 5 = 7$$

- $S_{\text{Ismail}} = \{3\}, S_{\text{Omar}} = \{1,2\}$

$$f_{\text{Ismail}}(\{3\}) + f_{\text{Omar}}(\{1,2\}) = 1 + 5 = 6$$

- $S_{\text{Ismail}} = \{1,2\}, S_{\text{Omar}} = \{3\}$

$$f_{\text{Ismail}}(\{1,2\}) + f_{\text{Omar}}(\{3\}) = 3 + 3 = 6$$

- $S_{\text{Ismail}} = \{1,3\}, S_{\text{Omar}} = \{2\}$

$$f_{\text{Ismail}}(\{1,3\}) + f_{\text{Omar}}(\{2\}) = 2 + 4 = 6$$

- $S_{\text{Ismail}} = \{2,3\}, S_{\text{Omar}} = \{1\}$

$$f_{\text{Ismail}}(\{2,3\}) + f_{\text{Omar}}(\{1\}) = 3 + 2 = 5$$

Here, the optimal distributions are $S_{\text{Ismail}} = \{1\}$ and $S_{\text{Omar}} = \{2,3\}$ or $S_{\text{Ismail}} = \{2\}$ and $S_{\text{Omar}} = \{1,3\}$, both with a total happiness of 7.

2.2.2 d-Concavity Proof

Notations : Let U a set of mathematical objects.

Proof of Characterization To prove a set function f over U is d-concave if and only if:

$$f(V \cup W) + f(V \cap W) \leq f(V) + f(W)$$

for all $V, W \subseteq U$.

Let $P(V, W, \{x\})$ the expression of the definition of a d-concave function.

Definition $\Rightarrow$ Characterization :

Let $V, W \in U$ with $V = \{x_1, x_2, \cdots, x_{|V|}\}$.
We consider for all $i \in \{1, 2, \cdots, |V| - 1\}$, $P(W, V \cap W, \{x_1\})$ and the inequalities $P(W \cup \{x_1, x_2, \cdots, x_i\}, (V \cap W) \cup \{x_1, x_2, \cdots, x_i\}, \{x_{i+1}\})$.
Summing in appropriate order and canceling terms from both sides, we get : $f(W \cup V) - f(W) \leq f((V \cap W) \cup V) - f(V \cap W)$. From which follows :

$$f(V \cup W) + f(V \cap W) \leq f(V) + f(W).$$

Characterization ⇒ Definition :
Since the characterization is symmetric over V, W we take $V := Z \cup \{x\}$ and $Z \subseteq W$, then we have :
$f(Z \cup W \cup \{x\}) + f((Z \cup \{x\}) \cap W) \leq f(Z \cup \{x\}) + f(W)$,
Thus, $f(W \cup \{x\}) + f(Z) \leq f(Z \cup \{x\}) + f(W)$.
Hence,

$$f(W \cup \{x\}) - f(W) \leq f(Z \cup \{x\}) - f(Z).$$

q.e.d.

Example Let $U = \{1, \ldots, 10\}$ and $f(V) = 1 - \frac{1}{|V|}$. Prove that f is d-concave.

Let V, W be subsets of U and v, w their respective cardinals and i, u respective cardinals of the intersection and the union, the problem is now equivalent to : $\frac{1}{u} + \frac{1}{i} \geq \frac{1}{v} + \frac{1}{w}$.
Since $g : x \to \frac{1}{x}$ is convex over R^+ and $i \leq min(v, w)$ and $u \geq max(u, v)$ and $u + i = v + w$ thus the desired result by Karamata's inequality. q.e.d.

Interpretation Explain the intuitive meaning of a d-concave function in the context of children's happiness.

Intuition behind d-concavity. The d-concavity of a set function f can be interpreted as a form of *diminishing marginal returns* applied to sets. In other words, adding the same element x to a larger (richer) set results in a smaller marginal gain than adding it to a smaller (poorer) set.

To illustrate, suppose that $f(S)$ measures the happiness of a child receiving a set of gifts S. If a child already has many gifts (W is large), giving them an additional gift (x) will not increase their happiness as much as giving a gift to another child with fewer gifts (V is small). This reflects the idea that the first gift is much more valuable than the tenth or twentieth.

Thus, d-concavity expresses a principle of fairness: it encourages distributions where marginal gains decrease as the size of the set increases.

Examples of d-concave functions. Here are three examples of d-concave functions:

A. $f(S) = \log(|S| + 1)$, where $|S|$ is the size of the set S. This function increases logarithmically, with diminishing returns as the set grows.

B. $f(S) = \max_{x \in S}\{x\}$, where $S \subseteq U$ is a subset of numbers.

   Proof : case work.

C. $f(S) = \frac{|S|}{|U|}$, where $|S|$ is the size of the set S. This function measures the proportion of the total set U that is included in S, with diminishing returns as the set size increases.

2.2.3 Matroid Examples

Uniform Matroid

5. Prove that $(U, \mathcal{I})$ is a matroid.

Proof :

- For (P1) : Given W, V sets such $V \in \mathcal{I} \Rightarrow |V| \leq k$, on the other hand$W \subseteq V \Rightarrow |W| \leq |V| \leq k \Rightarrow W \in \mathcal{I}$.
- For (P2) : Given $W, V \in \mathcal{I}$ and $|W| < |V| \Rightarrow |W| < k$, on the other hand $|W \cup \{x\}| \leq |W| + 1 \leq k \Rightarrow W \in \mathcal{I}$

Partition Matroid

6. Prove that $(U, \mathcal{I})$ is a matroid.

Proof :

- For (P1) : Given W, T sets such $W \in \mathcal{I}$, on the other hand $T \subseteq W$, $k_i \geq |W \cap V_i| \geq |T \cap V_i|$ for all $1 \leq i \leq l$, thus $T \in \mathcal{I}$.
- For (P2) : Given $W, T \in \mathcal{I}$ and $|W| < |T|$, let $i \in \{1, 2, \cdots, l\}$, if T, W aren't completely disjoint, choosing $\{x\} \in T \cap W$ would be enough. Else, WLOG we have $|T \cap V_i| = 0$ thus adding any element from W would work since it will get to $|W \cap V_i| \geq |\{x\} \cap V_i|$ and q.e.d.

2.3 Maximizing a d-Concave Function in a Uniform Matroid

Existence of Solution

Show that there exists an optimal solution S such that $|S^| = k$.

Proof :

- **Given:** f is a monotone d-concave function.
- **Assume:** We have an optimal solution S' such that $|S'| < k$.
- **Monotonicity of f:** Due to the monotonicity of f, adding any element $e \in U \setminus S'$ cannot decrease $f(S')$. Formally,

$$f(S' \cup \{e\}) \geq f(S').$$

- **Adding elements:** Therefore, we can always add elements to S' without decreasing the value of the objective function, until $|S^*| = k$.
- **Conclusion:** Since we can add elements to S' until $|S^*| = k$ and the value of f does not decrease, there must exist an optimal solution with exactly k elements.

2.4 Greedy Algorithm Analysis

Greedy Algorithm Suboptimality

Demonstrate that the greedy algorithm does not always produce an optimal solution.

S	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{1,2\}$	$\{1,3\}$	$\{1,4\}$	$\{2,3\}$	$\{2,4\}$	$\{3,4\}$
$f(S)$	20	18	18	18	25	25	25	30	30	30

This is an example of a uniform matroid have an optimal solution $S^{=}\{2,3\}$ meanwhile the greedy algorithm chooses the following set $S_G = \{1,2\}$ as we observe that $f(S^{)} = 30 > 25 = f(S_G)$.

Performance Bound

2. Prove that :

$$\max_{e\in E}\left[f(S\cup\{e\}) - f(S)\right] \geq \frac{1}{k}\left[f(S^*) - f(S)\right].$$

Proof :
Let $S^* \setminus S = \{i_1, i_2, \ldots, i_p\}$, where $p \leq k$. Then we have:

$$\begin{aligned}
f(S^{)} &\leq f(S^{\cup}S) && (1)\\
&= f(S) + \sum_{j=1}^{p}\left[f(S\cup\{i_1,\ldots,i_j\}) - f(S\cup\{i_1,\ldots,i_{j-1}\})\right]\\
&\leq f(S) + \sum_{j=1}^{p}\left[f(S\cup\{i_j\}) - f(S)\right] && (2)\\
&\leq f(S) + \sum_{j=1}^{p}\max_{e\in E}\left[f(S\cup\{e\}) - f(S)\right] && (3)\\
&= f(S) + k\max_{e\in E}\left[f(S\cup\{e\}) - f(S)\right]. && (4)
\end{aligned}$$

Here, (1) follows from the monotonicity of f, (2) from the d-concavity of f, and (3) from maximality of the expression, and (4) from $p \leq k$.

▷ This will be denoted by **Lemma 1** in the next few proofs.

3. Conclude that :

$$f(S^{)} - f(S_{t+1}) \leq \left(1 - \frac{1}{k}\right)\cdot\left(f(S^{)} - f(S_t)\right),$$

moreover,

$$f(S_k) \geq \left(1 - \frac{1}{e}\right)\cdot f(S^*).$$

Proof :
Let S_t denote the solution of the greedy algorithm at the end of iteration t. Then by **Lemma 1**:

$$\max_{i\in U}\left(f(S_t\cup\{i\}) - f(S_t)\right) = f(S_{t+1}) - f(S_t),$$

which implies:

$$\frac{1}{k}\left(f(S^*) - f(S_t)\right) \leq f(S_{t+1}) - f(S_t).$$

This further implies:

$$f(S^)) - f(S_{t+1}) \leq \left(1 - \frac{1}{k}\right)\left(f(S^)) - f(S_t)\right).$$

By recursion:

$$f(S_k) \geq \frac{f(S^*)}{k}\left[1 + \left(1 - \frac{1}{k}\right) + \left(1 - \frac{1}{k}\right)^2 + \cdots + \left(1 - \frac{1}{k}\right)^k\right] + f(\emptyset).$$

Simplifying further:

$$f(S_k) \geq \left(1 - \left(1 - \frac{1}{k}\right)^{k+1}\right) f(S^)) \geq \left(1 - \frac{1}{e}\right) f(S^)).$$

Here, the last inequality uses $(1 - x \leq e^{-x})^*$ for all $x \in R$.
* : after getting $\left(1 - \frac{1}{k}\right)^{k+1} \leq (e^{-\frac{1}{k}})^{k+1} \leq e^{-1}$.

2.5 More Maximisation

Algorithm Description

Describe the greedy algorithm for maximizing a d-concave function subject to matroid constraint.

- **Initial state :** $S_0 = \emptyset, i = 0$.
- **Step 1** : Iterate over $e \in U \setminus S_i$, and take $e' = arg(\ max(f(S_i \cup \{e\}) - f(S_i))$ and $S_i \cup \{e'\} \in \mathcal{I}$
- **Step 2** : If such e' exists : $S_{i+1} := S_i \cup \{e'\}$ and $i := i + 1$ and return to **Step 1**.
- Terminate.

Performance Bound

Prove that:

$$f(S_G) \geq \frac{1}{2} f(S^*)$$

Proof : Let $S^* = \{a_1, a_2, \cdots, a_k\}$, and $S_G = \{b_1, b_2, \cdots, b_k\}$ in a way $S_i = \{b_1, b_2, \cdots, b_i\}$ the answer the Greedy constructs after exactly i iterations. We use now $(P2)$ of a matroid to construct a map $\Phi : \{1, 2, \cdots, k\} \to \{1, 2, \cdots, k\}$ in such a way $(S_G \setminus \{b_i\}) \cup \{a_{\Phi(i)}\} \in \mathcal{I}$ for all i. At step $i + 1$, by construction $a_{\Phi(i+1)}$ is available same as b_{i+1} then we obtain :
$\Delta(GRD, i+1) = f(S_i \cup b_{i+1}) - f(S_i) \geq f(S_i \cup a_{\Phi(i+1)}) - f(S_i) = \Delta(OPT, i+1)$. The minorization then is :

$$f(S_G) = \sum_{i=0} \Delta(GRD, i + 1)$$

$$f(S_G) \geq \sum_{i=0} \Delta(OPT, i + 1)$$

We now propose an equivalent expression to the definition of d-concavity.

Lemma 2 : For any two sets, $W \subseteq V \subseteq U$:

$$f(V) \leq f(W) + \sum_{e \in V \setminus W} f(V \cup \{e\}) - f(V)$$

By **Lemma 2 :**

$$f(S_G) \geq f(S_G \cup S^*) - f(S_G)$$

$$f(S_G) \geq \frac{1}{2} f(S^*)$$

Where the penultimate inequality is a consequence of d-concavity, and the last one follows by the fact f is nondecreasing.

2.6 Revisiting the Gift Distribution Problem

2.7 Formulation as a Matroid Problem

Formulate the gift distribution problem as a maximization problem under a partition matroid.

Prior to formulating the problem appropriately we state the following theorem. **Theorem 1 :** *Given f and g two d-concave functions $f + g$ is d-concave.*

Formulation :

- **Universe Set :** $U := \mathcal{N}$
- **D-concave Function :** By **Theorem 1**, the target function to maximize f is d-concave.
- **Matroid :** Since the aim of the function is to apply theoretical approach to an intuitive real life scenario, we choose to pick a partition matroid where the integers bounding values are picked on some real life preferences as an abstraction we consider it an arbitrary sequence of positive integers.

2.8 Fairness Discussion

Discuss cases where this distribution method may be unfair. Propose alternative formulations and analyze simple algorithms (e.g., greedy) for these formulations.

This last question will be discussed as questions and motivations and some (not always formal) ideas.

2.8.1 Why do we have an increasing hypothesis?

An intuitive look shows that more gifts can't always be the reason of more satisfaction, thus it is an interesting pursuit to think about non monotonic d-concave functions.

Observation 1 : a Greedy Algorithm can't α-approximate this problem. Consider the function f :

$$f(S) = \begin{cases} |S|, & \text{if } a_i \in S. \\ 505, & \text{else.} \end{cases}$$

easily you can remark since ties are broken arbitrarily we can't approximate an optimal solution (505 is only a joke regarding that one Arctic Monkeys song, put any scalar there).
Motivation 1 : Stack overflow post about approximation algorithms and expected value. Link.
Now with the tool of expected value in our hand, let's try to bound the Random algorithm performance (note that after some searching we found prior work that solved the problem with much better bounds than us).
Algorithm 1 : Intuitive Random Algo.

Algorithm 1 Some d-concave f

Let $X_0 \leftarrow \emptyset, Y_0 \leftarrow U$.
for $i = 1$ to n **do**
    with probability $1/2$ do $X_i \leftarrow X_{i-1} + u_i, Y_i \leftarrow Y_{i-1}$.
    otherwise $X_i \leftarrow X_{i-1}, Y_i \leftarrow Y_{i-1} - u_i$.
return X_n (or equivalently Y_n).

Claim 1 : $E[f(X_n)] \geq \frac{1}{4} f(OPT)$ with usual notations.
Given the motivation from previous bounds we always tend to create telescopic sums, so we recall a lemma used to prove a $\frac{1}{2}$-approximation for another randomized greedy.
Lemma 3 : *For every* $1 \leq i \leq |U|$ *:*

$$E[f(OPT_{i-1}) - f(OPT_i) + f(\overline{OPT}i - 1) + f(\overline{OPT}_i) \leq f(X_i) - f(Xi - 1) + f(Y_i) - f(Y_{i-1})$$

We present a proof from a previous paper mentionned since our work was to claim the bound.
https://theory.epfl.ch/moranfe/Publications/FOCS2012.pdf
Proof of Lemma 3 : Notice that it suffices to prove Inequality (1.12) conditioned on any event of the form $X_{i-1} = A_{i-1}$ for which the probability that $X_i - 1 = A_i - 1$ is non-zero (notice that this can only happen when $A_{i-1} \subseteq \{u_1, u_2, \ldots, u_{i-1}\}$). Fix such an event. The rest of the proof implicitly assumes everything is conditioned on this event. Note that, due to the conditioning, the random variables Y_{i-1}, OPT_{i-1} and OPT_{i-1} become deterministic.
If $u_i \notin OPT$ (and, hence, $u_i \in \overline{OPT}$), then

$$\begin{aligned} &E[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] \\ &= \frac{1}{2}\left[f(X_{i-1} + u_i) - f(X_{i-1}) + f(Y_{i-1} - u_i) - f(Y_{i-1})\right] \\ &\geq \frac{1}{2}\left[f(OPT_{i-1}) - f(OPT_{i-1} - u_i) + f(OPT_{i-1}) - f(OPT_{i-1} + u_i)\right] \\ &= E[f(OPT_{i-1}) - f(OPT_i) + f(OPT_{i-1}) - f(OPT_i)]. \end{aligned}$$

To see that the inequality follows by d-concavity notice that $X_{i-1} \subseteq [(OPT \cup X_{i-1}) \cap Y_{i-1}] - u_i = OPT_{i-1} - u_i$ and $OPT_{i-1} = (OPT \cup X_{i-1}) \cap Y_{i-1} \subseteq Y_{i-1} - u_i$ (since $u_i \notin OPT$).
The proof that the lemma holds also when $u_i \in OPT$ is similar, and is, therefore, omitted.

Proof : The rest is very simple, summing up and using the fact the sum is telescopic we get :

$$E[f(\overline{\mathrm{OPT}}_0) - f(\mathrm{OPT}_n) + f(\overline{\mathrm{OPT}}_n) - f(\overline{\mathrm{OPT}}_0)] \leq E[f(X_n) - f(X_0) + f(Y_n) - f(Y_0)]$$

with conventional notations. We assume $f \geq 0$ and recalling necessary values. We obtain :

$$E[f(X_n)] = E[f(Y_n)] \geq \frac{1}{4}\left(f(\mathrm{OPT}) + f(\overline{\mathrm{OPT}}) + f(\emptyset) + f(\mathcal{N})\right) \geq \frac{1}{4} \cdot f(\mathrm{OPT})\,.$$

It is worth the mention that in a particular case of d-concave non-monotone symmetric functions i.e $f(A) = f(\overline{A}) \Rightarrow f(OPT) = f(\overline{OPT})$ the bound can be further optimized to $\frac{1}{2}$-approximation.

3 Painville

3.1 Introduction

This work addresses the problem of redistributing bread on graphs, where each vertex represents a villager with an associated balance function. The goal is to determine if a given balance configuration can be transformed into a feasible state where all balances are non-negative, using local operations that respect the graph's structure.

We formulate the problem on a graph $G = (V, E)$, where each vertex v has a balance $S(v)$. Operations involve vertices giving or receiving bread based on their neighbors' balances. The main challenge is to find a sequence of operations that ensures all balances are non-negative.

Key claims include that a balance configuration is resolvable if and only if the sum of all balances is non-negative. We also conjecture that if the total bread is greater than or equal to the graph's Betti number, it is always possible to resolve the configuration with at most one negative balance. These claims guide the development of algorithms and lead to conjectures that connect the structure of the graph to the feasibility of balance resolution.

3.2 Mathematical Model

3.2.1 Question 1

1. Definition of the balance function S A balance function S on a graph $G = (V, E)$ is a function:

$$S : V \to Z$$

where $S(v)$ represents the bread balance (positive or negative integer) of villager v.

2. Effects of giving/receiving bread When villager v asks for bread from their neighbors:

- $S(v)$ increases by $\deg(v)$, where $\deg(v)$ is the degree of vertex v (i.e., the number of edges connected to v).
- For each neighbor u of v, $S(u)$ decreases by the number of edges between u and v.

When villager v gives bread to their neighbors:

- $S(v)$ decreases by $\deg(v)$.
- For each neighbor u of v, $S(u)$ increases by the number of edges between u and v.

3. Mathematical Problem Given an initial balance function S_0, find a sequence of giving/receiving operations that transforms S_0 into a balance function S where:

$$\forall v \in V,\ S(v) \geq 0$$

In other words:

- **Input:** Graph $G = (V, E)$ and initial balance function S_0.
- **Goal:** Find a sequence of operations (if one exists) to reach S where $S(v) \geq 0$ for all $v \in V$.
- **Allowed Operations:** For any vertex v, you can either:
  - (a) Give 1 bread to all neighbors.
  - (b) Receive 1 bread from all neighbors.

A balance function S is resolvable if there exists a sequence of valid operations that can transform it into a function where all vertices have non-negative values.
The set of all resolvable balance functions on graph G is denoted:

$$\mathrm{Sol}(G)$$

This formulation captures:

- The discrete nature of bread distribution (integer values).
- The local nature of transactions (only between connected vertices).
- The simultaneity of giving/receiving (affects all neighbors at once).
- The goal of avoiding negative balances.

3.2.2 Question 2

We consider K_3 as our graph below.

Example 1 - Charity Operation (C_v)

- **Initial state:** $S(v_1) = 1, S(v_2) = -1, S(v_3) = 0$.
- **Apply C_1 (charity from vertex 1):**
  - v_1 gives 1 bread to each neighbor.
  - v_1's balance decreases by 2 (degree = 2): $1 \to -1$.
  - v_2's balance increases by 1: $-1 \to 0$.
  - v_3's balance increases by 1: $0 \to 1$.

Example 2 - Begging Operation (M_v)

- **Initial state:** $S(v_1) = -1, S(v_2) = 3, S(v_3) = 1$.
- **Apply M_1 (begging from vertex 1):**
  - v_1 receives 1 bread from each neighbor.
  - v_1's balance increases by 2 (degree = 2): $-1 \to 1$.
  - v_2's balance decreases by 1: $3 \to 2$.
  - v_3's balance decreases by 1: $1 \to 0$.

Key Observations

- Each operation affects all neighbors simultaneously.
- The effect on the initiating vertex is proportional to its degree.
- Operations can sometimes create new negative balances while fixing others.
- The challenge is finding the right sequence of operations to make all balances non-negative.

3.2.3 Question 3

1. Analysis of Operations In both cases C_v (charity) and M_v (begging), bread is redistributed among the neighbors. The total change in bread across the graph for each operation is zero (conservation property).

2. Sufficient Condition for Resolubility A balance function S is not resolvable if:

$$\sum_{v \in V} S(v) < 0$$

Proof of this sufficient condition:

- Suppose the initial sum is negative: $\sum_{v \in V} S_0(v) < 0$.
- Each operation (whether C_v or M_v) preserves this sum.
- The goal is to achieve $S(v) \geq 0$ for all v.
- If we succeeded in achieving this, the sum of balances would be ≥ 0.
- This would contradict the initial assumption that the sum is negative.
- Therefore, if the sum is negative, S cannot be resolved.

3.3 Operations on Balance Functions.

A. $F(S)$ is solvable if and only if S is solvable, since if $F(S)$ is solvable then just apply F to S and then use the appropriate sequence to solve $F(S)$ which means S is solvable. OTOH, if S is solvable, let $\overline{F}$ because a sequence of moves that cancel F, an intuitive example if we consider a binary representation of F as a string $T = (a_1, a_2, \cdots, a_n)$ when $a_i = 1$ if we perform a **lending** move, and 0 otherwise. $\overline{F}$ would suggest a string $\overline{T} = (a_n \oplus 0, a_{n-1} \oplus 0, \cdots, a_1 \oplus 0)$, one can easily see if S is solvable, then apply $\overline{F}$ to $F(S)$ and you get S then just solve S.
The following algorithm work, let a **borrowing** move applied to vertex v we want $S(v) := S(v) + deg_G(v)$ and if $v_i v \in E$ then $S(v_i) := S(v_i) - 1$ after some sequence of **lending** moves, now for every vertex in $V \setminus \{v\}$ apply a **lending** move, non-adjacent vertices to v have a net gain of 0 which is GOOD, neighbors of v give $deg_G(v_i)$ breads then gain from their neighbors different than v a bread which is GOOD, and v gains $deg_G(v)$ which is GOOD. q.e.d. (This proof stay analogous if we allow multi-edges and is therefore, omitted).

B. Let W be the sum of bread balance in G. W is invariant under **lending** moves, then if we know exactly $|G| - 1$ elements from S then the last element is uniquely defined. Thus the desired bijection comes intuitively.
Bijection suggestion : Let σ_F be a balance function such $F(\sigma_F)$ has only non-negative elements

C. $S_{sol(\sigma_F)} = -S_F + \Delta$ where Δ is a balance function that has only nonnegative elements, note that it is easy to find the existence of some Δ that would secure the bijectivity which justify 2.

3.4 Linear Equations

A. The matrix M then would be indicating, by rows since it is symmetric over the main diagonal, how much breads a vertex will get after performing some operation F, to quantify we get as an example f an operation of **lending** over vertex v_1, in the example graph below, $S(v_i) := M_{1,i} + S(v_i)$.

B. Unsolved

C. Unsolved

3.5 Algorithm

A. First example, won't work since we are unable to escape having a negative balance. Second example, work as indicated.

B. We recall the proof in **Operations over balance functions Question 2,** but this an analogous proof to show that a valid sequence of moves that solve S can be only **borrowing** moves (Proof is omitted since it's the exact same proof). Now we prove a helping lemma.

Lemma 1 : ***Borrowing*** *moves are commutative. Thus the order doesn't matter.*
Proof : WLOG, Let (M_1) and (M_2) be **borrowing** moves on vertices 1 and 2, let's analyse the resulting balance function, we remark that that vertices touched by the moves are exactly the set $Q = N(1) \cup N(2) \cup \{1\} \cup \{2\}$, one can remark that vertices in Q change by $S(v_i) := S(v_i) + M_{1,i} + M_{2,i}$ in the language of the matrix in **Linear Equations** section, since the change would be symmetric if we flip the vertices 1 and 2, and Q is symmetric over them, we conclude commutativity.

Now with **Lemma 1** in hand we reorder the **borrowing** moves such that we perform the **borrowing** move on vertices $1, 2, \cdots, n$ in this order successively and one move per vertex, left operations can be concatenated in any order, by the proof of **Operations over balance functions Question 2,** the first n moves would cancel out, thus considering the minimal begging algorithm, we get a contradiction. q.e.d.

C. This is exactly a corollary of **Lemma 1**.

D. Not always, since one can consider $G = K_3$ with $S = (-1, 0, 1)$.

3.6 W-Stable Balance Functions

This part will an open discussion of how we are trying to find a solution to third question. In other words, bunch of lemmas, ideas, proofs, and a final conclusion.

Motivation 1 : Since we delete edges and try to look at resolvability of S, a natural quantity to consider is the the Betti number $B(G) = E - V + 1$ of the graph.

3.6.1 Specific cases

$B = 0$, then G is a tree, we claim that if

$$\sum_{v \in V} S(v) \geq B = 0$$

then S is solvable.

Proof :

Lemma : We can move any amount of bread from a vertex to another in a tree with no change in other vertices.

Lemma's Proof : Consider a tree T with a bread configuration $S(G)$. Assume there is one bread initially at vertex x, which we aim to transfer to vertex y. Since T is a tree, there is a unique path $P = x, x_1, x_2, \ldots, x_{n-1}, y$ connecting x to y, where n is the distance between x and y. Define the distance between x and any vertex x_i along P as i.

We proceed as follows:

- For each vertex x_i along the path P, the x_i-subtree will borrow i breads.
- At vertex y, the y-subtree borrows n breads, where n is the distance to x.

Next, we analyze how the bread amounts at vertices change after applying these moves:

A. **Case 1:** When $v = x$, vertex x has transferred out exactly one bread. Thus, its net change is a decrease of one bread.

B. **Case 2:** If $v = x_i$ (where x_i is any intermediate vertex on P), vertex x_i borrows $\lceil d(x_i) \rceil i$ breads. Simultaneously, the vertex provides $\lceil d(x_i) - 2 \rceil i + (i-1) + (i-2)$ breads, which simplifies back to $\lceil d(x_i) \rceil i$. Hence, the total change for x_i is zero.

C. **Case 3:** For any vertex v located in the subtree of x_i, vertex v borrows $\lceil d(v) \rceil i$ breads. At the same time, an equal $\lceil d(v) \rceil i$ is borrowed back, yielding a net change of zero.

D. **Case 4:** When $v = y$, vertex y borrows $\lceil d(y) \rceil n$ breads. However, it also provides $(n-1) + \lceil d(y) - 1 \rceil n$ breads, which reduces to $\lceil d(y) \rceil n - 1$. The net effect for y is therefore an increase of precisely one bread.

As a result, the one bread originally placed at vertex x has successfully been moved to vertex y. The bread count for all other vertices remains unaffected by this transfer. q.e.d.

Our result then follows immediately since it becomes a matter of discrete continuity that implies each value can have either to floor of the mean $S(v)$ or its ceil. G **is a cycle**

Sketch of idea

Let C be a cycle with a bread configuration $S(C)$, where the total number of breads on the cycle is one. The goal is to move positive breads to a negative vertex without creating any new negative vertices.

A. **Initial Setup:**
   - Assume there exists a negative vertex v.
   - Orient the cycle such that v is at the bottom position (approximately the six o'clock position).
   - Define:
     - x: the closest positive vertex to v in the clockwise direction.
     - y: the closest positive vertex to v in the counterclockwise direction.
   - If $x = y$, then vertex $x = y$ must have at least 2 breads, since the total number of breads is one.

B. **Paths:**
   - Let P_1 be the path $x = x_0, x_1, x_2, \ldots, x_n = v$ (the clockwise path to v).
   - Let P_2 be the path $y = y_0, y_1, y_2, \ldots, y_m = v$ (the counterclockwise path to v).
   - Assume without loss of generality that $n \leq m$, where n and m are the lengths of P_1 and P_2, respectively.

C. **Bread Lending Rules:**
   - Vertices on the (x, y)-path that do not include v lend n breads to their neighbors.
   - For each i:
     - Vertex x_i lends $n - i$ breads.
     - Vertex y_i lends $(n - i)$ breads, unless $(n - i)$ is negative, in which case the vertex lends nothing.

D. **Case Analysis:**

E. **Case 1:** $w = x$ or $w = y$. Vertex w lends n breads to its two neighbors but receives n and $n - 1$ breads back. Net change: -1 bread.

F. **Case 2:** w is a vertex on the (x, y)-path not including v. Vertex w lends n breads to its neighbors but receives the same amount back. Net change: 0 breads.

G. **Case 3:** w lies on P_1, say $w = x_k$. Vertex w lends $n - k$ breads to each neighbor but gains $n - k + 1$ and $n - k - 1$ breads back. Net change: 0 breads.

H. **Case 4:** $w = v$ or w lies on P_2 as the first vertex that does not lend. Vertex w receives one bread from its neighbor and lends nothing. Net change: $+1$ bread. If v is the first non-lending vertex for both P_1 and P_2, v gains 2 breads.

I. **Case 5:** w lies on P_2, does not lend, and receives nothing. Net change: 0 breads.

Conclusion: Positive breads can be moved to the negative vertex v without creating any additional negative vertices. q.e.d.

General conjecture of sufficient condition

Conjecture 1. *Given a connected graph G with dollar configuration $S(G)$, if the total number of breads on the graph is equal to or exceeds the Betti number of the graph, then there is a sequence of borrowing moves that leads to at most one negative vertex on the graph.*